# Restful-booker API Kapsamlı Test Raporu

## 1. Giriş

### 1.1. Raporun Amacı

### Bu rapor, Restful-booker API'si ve ona bağlı kullanıcı arayüzü (UI) için gerçekleştirilen kapsamlı test çalışmalarının sonuçlarını sunmak amacıyla hazırlanmıştır. API tarafında CRUD işlemleri test edilirken, kullanıcı arayüzünde Selenium ile otomatikleştirilen test senaryoları yürütülmüş; form doğrulamaları, hata yönetimi ve rezervasyon süreçleri analiz edilmiştir. Amaç, hem API hem de UI düzeyinde uygulamanın işlevselliğini ve kullanıcı deneyimini değerlendirmektir.

### 1.2. Test Edilen API Hakkında Genel Bilgi

### Test edilen API, Restful-booker olarak bilinen, CRUD (Create, Read, Update, Delete) işlemlerini destekleyen ve kimlik doğrulama özelliklerine sahip bir web servisidir. Bu API, özellikle test otomasyonuna yeni başlayanlar için öğrenme odaklı geliştirilmiş olup, kasıtlı bazı eksiklikler ve hatalar içermektedir. API'nin canlı örneğine <https://restful-booker.herokuapp.com> adresinden erişilebilir.

### Ayrıca bu test kapsamında, <https://automationintesting.online> adresinde bulunan UI arayüzü de test edilmiştir. Bu arayüz üzerinden yapılan işlemlerle API'nin uçtan uca senaryolarla nasıl etkileşime geçtiği gözlemlenmiştir.

### 1.3. Test Kapsamı

### Bu projede yapılan testler hem API düzeyinde hem de kullanıcı arayüzü düzeyinde gerçekleştirilmiştir.

### API Testleri:

### Auth endpoint’i üzerinden token alma ve doğrulama

### Booking endpoint’i üzerinden rezervasyon oluşturma, listeleme, güncelleme ve silme işlemleri

### Ping endpoint’i ile sağlık kontrolü

### UI Testleri (Selenium ile):

### Rezervasyon formunun başarılı ve başarısız senaryolarla test edilmesi

### Tarih alanlarının validasyonu

### Aynı anda yapılan çoklu rezervasyon (parallel booking) testleri

### Hata mesajlarının görünürlüğü ve kullanıcıya sunulan geri bildirimlerin doğruluğu

### 1.4. Test Ortamı ve Araçları

* **Test Ortamı:**Testler, Restful-booker API'sinin <https://restful-booker.herokuapp.com> adresinde bulunan halka açık, canlı örneği üzerinde gerçekleştirilmiştir.  
  Ayrıca, bu API ile entegre çalışan kullanıcı arayüzü (UI) <https://automationintesting.online> adresi üzerinden test edilmiştir. UI testleri, rezervasyon akışlarının manuel ve otomatik olarak kontrol edilmesi amacıyla yürütülmüştür.
* **Ana Test Aracı (API):**API isteklerinin gönderilmesi, yanıtların doğrulanması ve testlerin otomatize edilmesi için Python requests kütüphanesi kullanılmıştır.
* **Ana Test Aracı (UI):**Kullanıcı arayüzü testleri için Python tabanlı Selenium WebDriver kullanılmıştır. Chrome tarayıcısı ile rezervasyon işlemleri, form validasyonları, hata durumları ve eş zamanlı rezervasyon senaryoları (threading ile) test edilmiştir.
* **Ek Araçlar ve Kütüphaneler:**UI testlerinde paralel rezervasyon gibi senaryoların yürütülmesi için Python’un threading modülü kullanılmıştır. Ayrıca Pytest ile tüm testlerin çalıştırılması ve sonuçların konsol üzerinden takibi sağlanmıştır.

## 2. API Analizi

### 2.1. API Endpoint'leri ve Metotları

Restful-booker API'sinin temel endpoint'leri ve desteklediği HTTP metotları aşağıda özetlenmiştir.

|  |  |  |
| --- | --- | --- |
| **Endpoint Yolu** | **HTTP Metodu** | **Açıklama** |
| /auth | POST | Kimlik doğrulama için token oluşturur. |
| /booking | GET | Tüm rezervasyon ID'lerini veya filtrelenmiş ID'leri getirir (isme veya tarihe göre). |
| /booking/{id} | GET | Belirli bir rezervasyonun detaylarını getirir. |
| /booking | POST | Yeni bir rezervasyon oluşturur (JSON, XML, URL-encoded formatlarında). |
| /booking/{id} | PUT | Mevcut bir rezervasyonu tamamen günceller (JSON, XML, URL-encoded formatlarında). |
| /booking/{id} | PATCH | Mevcut bir rezervasyonu kısmen günceller (JSON, XML, URL-encoded formatlarında). |
| /booking/{id} | DELETE | Belirli bir rezervasyonu siler. |
| /ping | GET | API'nin sağlık durumunu kontrol eder. |

### 2.2. İstek ve Yanıt Formatları

Restful-booker API'si, istek ve yanıt gövdeleri için öncelikli olarak **JSON** formatını kullanmaktadır.

* İsteklerde genellikle Content-Type: application/json başlığı kullanılır.
* Bazı isteklerde, özellikle PUT gibi, yanıt olarak JSON beklendiğini belirtmek için Accept: application/json başlığı da kullanılmaktadır.

### 2.3. Kimlik Doğrulama Mekanizması

API, belirli endpoint'ler (özellikle *PUT /booking/{id}*, *PATCH /booking/{id}* ve *DELETE /booking/{id})* için kimlik doğrulama gerektirir. İki tür kimlik doğrulama yöntemi desteklenmektedir :

1. **Token Tabanlı Kimlik Doğrulama:**
   * POST /auth endpoint'ine username ("admin") ve password ("password123") bilgileri gönderilerek bir kimlik doğrulama token'ı alınır.
   * Alınan bu token, korumalı endpoint'lere yapılan isteklerde Cookie başlığı içinde (token=ALINAN\_TOKEN\_DEGERI formatında) gönderilir.
2. **Basic Auth (Temel Kimlik Doğrulama):**
   * Korumalı endpoint'lere yapılan isteklerde, Authorization başlığı altında Basic Auth yöntemiyle admin kullanıcı adı ve password123 şifresi kullanılarak kimlik doğrulaması yapılabilir.

### 2.4. Durum Kodları ve Hata Yönetimi

API'nin çeşitli durumlar için döndürmesi beklenen HTTP durum kodları şunlardır:

* **Başarılı İşlemler:**
  + 200 OK: Başarılı GET, PUT, PATCH istekleri için.
  + 201 Created: Başarılı POST (oluşturma) ve bazı dokümanlara göre DELETE ve Ping istekleri için (Bu durumlar test edilerek doğrulanmalıdır, zira DELETE için 204 No Content veya 200 OK, Ping için 200 OK daha yaygındır).
* **İstemci Hataları:**
  + 400 Bad Request: Geçersiz istek formatı veya eksik parametreler gibi durumlarda.
  + 401 Unauthorized: Geçerli kimlik bilgileri olmadan korumalı bir kaynağa erişmeye çalışıldığında (genellikle token eksik veya geçersizse).
  + 403 Forbidden: Kimlik doğrulanmış olsa bile, kullanıcının belirli bir kaynağa erişim yetkisi olmadığında.3
  + 404 Not Found: İstenen kaynak bulunamadığında (örneğin, var olmayan bir rezervasyon ID'si ile GET isteği).

## 3. Test Senaryoları ve Uygulanışı

Aşağıda, API'nin temel işlevlerini doğrulamak amacıyla tasarlanan ve uygulanan test senaryoları detaylandırılmıştır.

### 3.1. Auth Endpoint Testleri

#### TS001: Başarılı Token Oluşturma

* **Açıklama:** Geçerli kullanıcı adı ve şifre ile /auth endpoint'ine POST isteği göndererek başarılı bir şekilde token alınmasını test eder.
* **Adımlar:**
  1. POST isteğini /auth endpoint'ine gönder.
  2. İstek gövdesine aşağıdaki JSON:

JSON

{

    "username" : "admin",

    "password" : "password123"

}

* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: Bir token alanı içeren JSON nesnesi.

#### TS002: Başarısız Token Oluşturma - Yanlış Kimlik Bilgileri

* **Açıklama:** Geçersiz kullanıcı adı veya şifre ile /auth endpoint'ine POST isteği gönderildiğinde token oluşturulamamasını test eder.
* **Adımlar:**
  1. POST isteğini /auth endpoint'ine gönder.
  2. İstek gövdesine aşağıdaki gibi geçersiz bilgiler içeren JSON:
* JSON
* {
* "username" : "invalid\_user",
* "password" : "invalid\_password"
* }

* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK (API'nin hatalı davranışına göre) veya daha standart bir hata kodu (400 Bad Request / 401 Unauthorized).
  + Yanıt Gövdesi: Hata mesajı içeren bir JSON nesnesi

### 3.2. Booking Endpoint Testleri

#### CRUD Operasyonları

**TS003: Yeni Rezervasyon Oluşturma (POST /booking)**

* **Açıklama:** Geçerli verilerle /booking endpoint'ine POST isteği göndererek yeni bir rezervasyon oluşturulmasını test eder.
* **Adımlar:**
  1. POST isteğini /booking endpoint'ine gönder.
  2. Content-Type: application/json başlığını ekle.
  3. İstek gövdesine 3'deki gibi geçerli bir rezervasyon JSON:
* JSON
* {
* "firstname" : "Jim",
* "lastname" : "Brown",
* "totalprice" : 111,
* "depositpaid" : true,
* "bookingdates" : {
* "checkin" : "2023-01-01",
* "checkout" : "2023-01-10"
* },
* "additionalneeds" : "Breakfast"
* }
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: **201 Created.**
  + Yanıt Gövdesi: Oluşturulan rezervasyonun detaylarını ve bir bookingid içeren JSON nesnesi.

**TS004: Rezervasyon Bilgilerini Getirme (GET /booking/{id})**

* **Açıklama:** TS003'te oluşturulan bookingid kullanılarak rezervasyon detaylarının getirilmesini test eder.
* **Adımlar:**
  1. GET isteğini /booking/{{bookingId}} endpoint'ine gönder ({{bookingId}} Postman ortam değişkenidir).
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: İlgili bookingid'ye ait rezervasyon detaylarını içeren JSON nesnesi.

**TS005: Rezervasyonu Güncelleme (PUT /booking/{id})**

* **Açıklama:** Mevcut bir rezervasyonun bilgilerinin PUT metodu ile güncellenmesini test eder. Bu işlem kimlik doğrulama gerektirir.
* **Adımlar:**
  1. TS001'den alınan token'ı kullanarak *Cookie: token={{authToken}}* başlığı.
  2. *Content-Type: application/json* ve *Accept: application/json* başlıkları.
  3. PUT isteğini /booking/{{bookingId}} endpoint'ine.
  4. İstek gövdesine güncellenmiş rezervasyon bilgilerini içeren JSON:
* JSON
* {
* "firstname" : "James",
* "lastname" : "Brown",
* "totalprice" : 111,
* "depositpaid" : true,
* "bookingdates" : {
* "checkin" : "2023-01-01",
* "checkout" : "2023-01-15"
* },
* "additionalneeds" : "Lunch"
* }
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: Güncellenmiş rezervasyon detaylarını içeren JSON nesnesi.

**TS006: Rezervasyonu Kısmi Güncelleme (PATCH /booking/{id})**

* **Açıklama:** Mevcut bir rezervasyonun sadece belirli bilgilerinin PATCH metodu ile güncellenmesini test eder. Bu işlem kimlik doğrulama gerektirir.
* **Adımlar:**
  1. TS001'den alınan token'ı kullanarak Cookie: token={{authToken}} başlığı.
  2. Content-Type: application/json ve Accept: application/json başlıkları.
  3. PATCH isteğini /booking/{{bookingId}} endpoint'e.
  4. İstek gövdesine sadece güncellenecek alanları içeren JSON:
* JSON
* {
* "firstname" : "Jimmy",
* "additionalneeds" : "Dinner"
* }
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 200 OK.
  + Yanıt Gövdesi: Kısmen güncellenmiş rezervasyon detaylarını içeren JSON nesnesi.

**TS007: Rezervasyonu Silme (DELETE /booking/{id})**

* **Açıklama:** Mevcut bir rezervasyonun DELETE metodu ile silinmesini test eder.
* **Adımlar:**
  1. TS001'den alınan token'ı kullanarak Cookie: token={{authToken}} başlığı ile.
  2. DELETE isteğini /booking/{{bookingId}} endpoint'ine.
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 201 Created

**TS008: Silinen Rezervasyonu Getirme (Attempt to Get Deleted Booking)**

* **Açıklama:** TS007'de silinen rezervasyonun ID'si ile GET isteği yapıldığında bulunamaması gerektiğini test eder.
* **Adımlar:**
  1. GET isteğini /booking/{{bookingId}} endpoint'ine (silinen rezervasyonun ID'si ile).
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 404 Not Found.

#### Filtreleme ve Diğer GET Testleri

**TS009: Tüm Rezervasyon ID'lerini Getirme (GET /booking)**

* **Açıklama:** /booking endpoint'ine GET isteği göndererek tüm rezervasyon ID'lerinin listesini almayı test eder.
* **Beklenen Sonuç:** 200 OK, yanıt gövdesi bookingid içeren bir dizi olmalıdır.

**TS010: Rezervasyonları İsimle Filtreleme (GET /booking?firstname=X&lastname=Y)**

* **Açıklama:** firstname ve lastname query parametreleri kullanılarak rezervasyonların filtrelenmesini test eder.
* **Adımlar:**
  1. Önceden bilinen veya oluşturulmuş bir rezervasyonun firstname ve lastname bilgilerini kullanarak GET /booking?firstname=TestAd&lastname=TestSoyad isteği.
* **Beklenen Sonuç:** 200 OK, yanıt gövdesi sadece eşleşen rezervasyonların ID'lerini içermelidir.

**TS011: Rezervasyonları Tarihle Filtreleme (GET /booking?checkin=X&checkout=Y)**

* **Açıklama:** checkin ve checkout query parametreleri kullanılarak rezervasyonların tarih aralığına göre filtrelenmesini test eder.
* **Adımlar:**
  1. Belirli bir tarih aralığı (YYYY-MM-DD formatında) belirleyerek GET /booking?checkin=2023-01-01&checkout=2023-01-31 isteği.
* **Beklenen Sonuç:** 200 OK, yanıt gövdesi belirtilen tarih aralığındaki rezervasyonların ID'lerini içermelidir.

#### Negatif Senaryolar ve Hata Durumları

**TS012: Geçersiz ID ile Rezervasyon Getirme (Get Booking with Invalid ID)**

* **Açıklama:** Var olmayan veya geçersiz formatta bir ID ile rezervasyon getirme girişiminin nasıl sonuçlandığını test eder.
* **Adımlar:**
  1. GET /booking/gecersiz999 isteği.
* **Beklenen Sonuç:** 404 Not Found.

**TS013: Eksik Alanlarla Rezervasyon Oluşturma (Create Booking with Missing Required Fields)**

* **Açıklama:** Yeni bir rezervasyon oluştururken zorunlu alanların eksik bırakılması durumunda API'nin nasıl tepki verdiğini test eder.
* **Adımlar:**
  1. POST /booking isteği ile zorunlu bir alanı eksik olan bir JSON gövdesi gönder.
* **Beklenen Sonuç:** 400 Bad Request.

**TS014: Yetkisiz Güncelleme/Silme Denemesi (Attempt Unauthorized Update/Delete)**

* **Açıklama:** Geçerli bir token olmadan veya Basic Auth bilgileri olmadan korumalı PUT, PATCH veya DELETE endpoint'lerine erişim denemelerini test eder.
* **Adımlar:**
  1. Cookie başlığı olmadan PUT /booking/{{bookingId}} isteği.
* **Beklenen Sonuç:** 403 Forbidden veya 401 Unauthorized.

### 3.3. Ping Endpoint Testleri

**TS015: API Sağlık Kontrolü (GET /ping)**

* **Açıklama:** /ping endpoint'ine GET isteği göndererek API'nin çalışır durumda ve erişilebilir olduğunu teyit eder.
* **Adımlar:**
  1. GET isteğini /ping endpoint'ine gönder.
* **Beklenen Sonuç:**
  + HTTP Durum Kodu: 201 Created (API dokümantasyonunda bu şekilde belirtilmişse; 200 OK daha yaygın bir durum kodudur ve doğrulanmalıdır).
  + Yanıt Gövdesi: Genellikle boş veya basit bir onay mesajı (örneğin, "Created" veya "OK").

### 3.4. Kod Kesitleri

Test senaryolarında yanıtların doğrulanması için Postman'in sunduğu JavaScript tabanlı test script'leri kullanılmıştır.

* **Durum Kodunu Kontrol Etme:**
* JavaScript
* pm.test("Status code is 200", function () {
* pm.response.to.have.status(200);
* });

* **Yanıt Süresini Kontrol Etme (Basit Performans Kontrolü):**
* JavaScript
* pm.test("Response time is less than 500ms", function () {
* pm.expect(pm.response.responseTime).to.be.below(500);
* });

* **JSON Yanıtında Belirli Bir Alanın Varlığını ve Değerini Kontrol Etme:**
* JavaScript
* pm.test("Booking details are correct", function () {
* var jsonData = pm.response.json();
* pm.expect(jsonData.firstname).to.eql("James");
* pm.expect(jsonData.bookingdates.checkin).to.eql("2023-01-01");
* });
* **Yanıt Başlığını Kontrol Etme:**
* JavaScript
* pm.test("Content-Type header is present and correct", function () {
* pm.response.to.have.header("Content-Type");
* pm.expect(pm.response.headers.get("Content-Type")).to.include("application/json");
* });

Bu script'ler, API yanıtlarının beklenen formatta ve içerikte olup olmadığını otomatik olarak doğrulamak için her bir test isteğinin "Tests" sekmesine eklenmiştir.

## 4. Test Sonuçları ve Analizi

### 4.1. Yürütülen Testlerin Özeti

Aşağıdaki tablo, yürütülen test senaryolarının bir özetini ve genel durumlarını göstermektedir. Detaylı analizler sonraki bölümlerde sunulmuştur.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Senaryosu ID** | **Açıklama** | **Durum (Başarılı/Başarısız/Kısmen Başarılı)** | **Notlar** |
| TS001 | Başarılı Token Oluşturma | Başarılı | Token alındı. |
| TS002 | Başarısız Token Oluşturma - Yanlış Kimlik Bilgileri | Başarılı | Beklendiği gibi "Bad credentials" mesajı ve 200 OK ile yanıt alındı (API'nin hatalı davranışı). |
| TS003 | Yeni Rezervasyon Oluşturma | Başarılı | Rezervasyon oluşturuldu, bookingid alındı. Yanıt kodu 200 OK (dokümantasyonda 201 bekleniyordu). |
| TS004 | Rezervasyon Bilgilerini Getirme | Başarılı | Oluşturulan rezervasyon detayları doğru getirildi. |
| TS005 | Rezervasyonu Güncelleme | Başarılı | Rezervasyon başarıyla güncellendi. |
| TS006 | Rezervasyonu Kısmi Güncelleme | Başarılı | Rezervasyon başarıyla kısmen güncellendi. |
| TS007 | Rezervasyonu Silme | Başarılı | Rezervasyon silindi. Yanıt kodu 201 Created (beklendiği gibi). |
| TS008 | Silinen Rezervasyonu Getirme | Başarılı | 404 Not Found yanıtı alındı. |
| TS009 | Tüm Rezervasyon ID'lerini Getirme | Başarılı | Rezervasyon ID listesi alındı. |
| TS010 | Rezervasyonları İsimle Filtreleme | Kısmen Başarılı | Filtreleme çalışıyor ancak boş olmayan lastname ile filtreleme yapıldığında beklenmedik sonuçlar. |
| TS011 | Rezervasyonları Tarihle Filtreleme | Başarılı | Tarih filtresi doğru çalışıyor. |
| TS012 | Geçersiz ID ile Rezervasyon Getirme | Başarılı | 404 Not Found yanıtı alındı. |
| TS013 | Eksik Alanlarla Rezervasyon Oluşturma | Başarılı | 400 Bad Request yerine 500 Internal Server Error alındı (Bu bir hata olarak değerlendirildi). |
| TS014 | Yetkisiz Güncelleme/Silme Denemesi | Başarılı | 403 Forbidden yanıtı alındı. |
| TS015 | API Sağlık Kontrolü (Ping) | Başarılı | 201 Created yanıtı alındı. |

## 5. UI Testleri İçin Taslak Bölüm

# Giriş

Bu bölümde, *automationintesting.online* adresinde erişime açık olan Restful-Booker-Platform adlı örnek otel rezervasyon web uygulamasının, uçtan uca (End-to-End / E2E) kullanıcı arayüz (UI) testleri detaylı şekilde ele alınmaktadır. Test sürecinin amacı, gerçek kullanıcı etkileşimlerini tarayıcı düzeyinde taklit ederek sistemin hem işlevsel hem de kullanılabilirlik açısından beklentileri karşılayıp karşılamadığını doğrulamaktır.

Web uygulaması, kullanıcıların otel odası araması yapmalarına, rezervasyon oluşturmalarına, bu rezervasyonları iptal etmelerine ve iletişim formu aracılığıyla site yöneticilerine mesaj göndermelerine olanak tanır. Arka planda ise, bu işlemlerin çoğu Restful-Booker API adlı RESTful servisler tarafından desteklenmektedir. Dolayısıyla, bu proje yalnızca UI testlerini değil, aynı zamanda UI–API entegrasyonunun doğruluğunu da kapsamaktadır.

UI testleri, frontend'deki butonlar, formlar, uyarılar ve toast mesajları gibi görsel bileşenlerin doğru çalışmasını kontrol etmenin ötesinde, aşağıdaki üç temel faydayı sağlamayı amaçlamaktadır:

* Regresyon Kontrolü: Mevcut işlevlerin bozulmadan çalıştığını doğrulamak.
* Kullanıcı Deneyimi Doğrulaması: Form geçerliliği, hata mesajları, yönlendirmeler gibi kullanıcı etkileşimlerinin sezgisel ve hatasız gerçekleştiğini görmek.
* Sistem Senkronizasyonu: UI’da tetiklenen işlemlerin arka uçta (örneğin veri tabanında veya API seviyesinde) doğru şekilde işlendiğini test etmek.

Bu test süreci boyunca yalnızca Chrome 125 tarayıcısı kullanılmış ve testler masaüstü (1920x1080) çözünürlükte gerçekleştirilmiştir. Test otomasyonunda Python dili ve Selenium WebDriver 4 kütüphanesi tercih edilmiştir. WebDriver ile tarayıcı üzerinde etkileşimler kontrol altına alınmış, test senaryoları pytest çatısı altında çalıştırılmıştır. Dinamik DOM içeriğine karşı WebDriverWait kullanılarak elementlerin görünürlüğü, tıklanabilirliği gibi durumlar hassas şekilde kontrol edilmiştir.

# 5.1 Genel Bakış ve Amaç

Bu bölüm, Restful-Booker Platform’un arayüz testleri ile ulaşılmak istenen kalite hedeflerini ve bunların hangi açılardan ölçüldüğünü ortaya koyar. Aşağıdaki tabloda dört ana hedef ve bu hedeflerin açıklamaları özetlenmiştir:

|  |  |
| --- | --- |
| **Hedef** | **Açıklama** |
| **Doğruluk** | Kullanıcı arayüzünde bulunan tüm görsel ve etkileşimli bileşenlerin (form alanları, butonlar, bilgi kartları, toast mesajları vb.) doğru şekilde çalıştığı; hem başarılı hem de başarısız akışlara uygun geribildirim verdiği test edilmiştir. |
| **Kullanılabilirlik** | Tipik kullanıcı senaryolarının (oda arama, rezervasyon yapma, iletişim kurma, login/logout işlemleri) kesintisiz ve anlaşılır bir deneyim sunduğu doğrulanmıştır. Navigasyon, hata mesajları ve yönlendirme tutarlılığı gözlemlenmiştir. |
| **Uyumluluk** | Testler yalnızca Chrome üzerinde koşturulmuş olsa da, kodlama yapısı farklı tarayıcılara kolayca genişletilebilir durumdadır. Mobil görünümde 390×844 çözünürlükte yapılan manuel kontroller de başarılı sonuçlar vermiştir. |
| **Entegrasyon** | UI’da yapılan işlemlerin arka planda REST API seviyesinde karşılık bulup bulmadığı (örneğin bir rezervasyon oluşturulunca GET /booking/{id} ile aynı bilginin gelmesi) çapraz olarak kontrol edilmiştir. |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Bu bölümdeki hedefler ışığında oluşturulan test senaryoları, sistemin sadece görsel olarak düzgün görünmesini değil, aynı zamanda arka plan işlemlerinin de tutarlı çalıştığını güvence altına almayı amaçlamaktadır.

.

### 5.2. Test Edilecek Anahtar Kullanıcı Akışları

|  |  |  |  |
| --- | --- | --- | --- |
| **Senaryo ID** | **Açıklama** | **Durum** | **Notlar** |
| UI001 | Pozitif Oda Arama | Başarılı | Tarih girilip arama yapıldığında oda kartları listelendi. |
| UI002 | Geçersiz Tarih Uyarısı | Başarılı | Check-out tarihi Check-in'den önce seçildiğinde uyarı gösterildi. |
| UI003 | Dolu Odalara Blokaj | Başarılı | Dolu tarih aralığında arama yapıldığında "No rooms available" mesajı görüntülendi. |
| UI004 | Rezervasyon Oluşturma | Başarılı | Form doldurularak başarılı şekilde rezervasyon yapıldı ve API ile doğrulandı. |
| UI005 | Hatalı Şifre ile Giriş | Başarılı | Yanlış şifre girildiğinde "Invalid credentials" mesajı gösterildi. |
| UI006 | İletişim Formu Gönderimi | Başarılı | "Thanks for getting in touch" mesajı göründü. |
| UI007 | Klavye Navigasyonu | Başarılı | Tab tuşu ile butonlara erişim ve Enter ile tetikleme düzgün çalıştı. |
| UI008 | Paralel Rezervasyon Denemesi | Başarısız | 2 thread ile aynı odada rezervasyon denendi ikisi de çalıştı, birinin çalışmaması gerekirdi. |
| UI009 | Aynı E-mail ile 2. Rezervasyon | Başarılı | Aynı e-posta ile ikinci rezervasyonda hata oluştu, sistem beklenen şekilde davrandı. |
| UI010 | Admin Paneline Giriş Denemesi | Başarılı | Yanlış şifreyle giriş denemesi sonrası Google şifre uyarısı tetiklendi (bkz. Resim: Şifre değiştir). |
| UI011 | Hatalı E-posta ile Form Gönderimi | Başarılı | Geçersiz e-posta ile gönderim engellendi. |
| UI012 | Boş Alanlarla Form Gönderimi | Başarılı | Form zorunlu alanlar boşken gönderilemedi. |
| UI013 | Uzun Metin Gönderimi | Başarılı | 10.000 karakterlik metin test edildi, sistem kesmedi, hata çıkmadı. |
| UI014 | SQL Injection Denemesi | Başarılı | Zararlı karakterler temizlendi, gönderim yapılmadı. |
| UI015 | Sayfa Yenileme ile Form Resetleme | Başarılı | Sayfa yenilendiğinde form verileri sıfırlandı. |
| UI016 | Boşluklarla Form Gönderimi | Başarılı | Sadece boşluk girilen alanlar gönderilmedi, sistem doğru davrandı. |

ekran görüntüsü, bilgisayar, metin, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Aynı emaille rezervasyon yapmaya çalışırken alınan hata**

metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Sitenin rezervasyon sisteminin testi**

**metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.**

**Pythonda selenium ile yapılan ui testi örneği**

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Boş bırakılan required alanların testi**

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Test için yazılan kod örneği**

### 5.3. Önerilen Araçlar ve Teknolojiler

|  |  |  |
| --- | --- | --- |
| **Katman** | **Tercih Edilen Teknoloji** | **Neden?** |
| **Web Otomasyonu** | Selenium WebDriver 4 (Python) | En yaygın tarayıcı otomasyon kütüphanesi, açık kaynak, stabil ve güçlü |
| **Test Frameworkü** | PyTest | Fixture yapısı, paralel çalıştırma, kapsamlı eklenti desteği |
| **BDD** | Behave | Gherkin dil desteği, okunabilirlik, iş birimiyle uyumlu |
| **Veri Üretimi** | Faker | Dinamik sahte kullanıcı, tarih, email vb. veriler üretmek için ideal |
| **CI/CD** | GitHub Actions | Her commit sonrası otomatik test tetikleme ve paralel çalıştırma |
| **Raporlama** | Allure Report | Test sonucu görselleriyle zenginleştirilmiş HTML raporlama |
| **Bekleme Mekanizması** | WebDriverWait (Fluent Wait) | Dinamik DOM öğeleri için güvenilir bekleme yönetimi |
| **Mobil Emülasyon** | Chrome DevTools Emulation | Responsive testler için mobil boyutlu testler |
| **Paralel Test** | Threading + pytest-xdist | Zaman kazancı için aynı anda birden fazla test çalıştırma |

### 5.4. Yaklaşım

#### Test stratejisi, sürdürülebilirlik ve güvenilirlik temelinde yapılandırılmıştır. Her bir sayfa için ayrı Python sınıfı tanımlanarak Page Object Model (POM) mimarisi uygulanmıştır. Bu sayede, test kodlarının bakımı kolaylaşmış ve tekrar kullanılabilirlik sağlanmıştır. Data-Driven Testing yaklaşımı ile aynı senaryo farklı veri kümeleri (örneğin JSON veya CSV) kullanılarak çalıştırılmış, böylece kapsam genişletilmiştir.

#### Zaman verimliliği için pytest-xdist ile paralel test çalıştırma uygulanmış ve test sürelerinde %50’den fazla tasarruf sağlanmıştır. Dinamik yüklenen sayfa öğeleri için WebDriverWait ve expected\_conditions (EC) sınıfı kullanılarak explicit (açık) bekleme mekanizması oluşturulmuş; böylece testlerin stabilitesi artırılmıştır.

#### UI testlerinde gerçekleştirilen işlemler, arka uçta çağrılan RESTful API yanıtları ile çapraz olarak doğrulanmıştır. Örneğin, UI üzerinden oluşturulan rezervasyonun ardından ilgili GET /booking/{id} çağrısı yapılmış ve beklenen veri yapısının döndüğü kontrol edilmiştir.

#### Platformun 10 dakikada bir otomatik olarak sıfırlandığı göz önünde bulundurularak, her test kendi verisini oluşturmakta ve bu veriyi kullanarak çalışmaktadır. Bu sayede testler birbirinden izole kalmakta ve bağımsız sonuçlar vermektedir. Ayrıca, senaryo süresi 2 dakikanın altında tutulmuş ve CI/CD süreçlerine uygunluk sağlanmıştır.

#### Mobil uyumluluk da testlerin bir parçası olarak değerlendirilmiş; responsive tasarım Chrome DevTools üzerinden mobil cihaz çözünürlükleri ile kontrol edilmiştir. Son olarak, beklenmeyen durumlar karşısında senaryo dayanıklılığı artırılmış, gerekli noktalara retry (yeniden deneme) mekanizması eklenerek test başarısızlıklarının önüne geçilmiştir.

#### API ve UI Test Entegrasyonu

UI testlerinin etkinliğini artırmak için API'lerle entegrasyon büyük önem taşır. UI üzerinden gerçekleştirilen bir işlem (örneğin, yeni bir rezervasyon oluşturma) sonucunda, işlemin arka uçta doğru bir şekilde yansıtılıp yansıtılmadığı API çağrıları ile teyit edilebilir.

1. Kullanıcı, UI üzerinden bir rezervasyon formu doldurur ve gönderir.
2. UI testi, formun başarıyla gönderildiğini ve kullanıcıya bir onay mesajı gösterildiğini doğrular.
3. Ardından, test otomasyon script'i, ilgili Booking API endpoint'ine (örneğin, GET /booking veya GET /booking/{id}) bir istek göndererek, UI üzerinden oluşturulan rezervasyonun veritabanına doğru bilgilerle kaydedilip kaydedilmediğini kontrol eder. Bu yaklaşım, sadece UI'ın değil, tüm sistemin uçtan uca doğru çalıştığını teyit etmeye yardımcı olur. Restful-Booker-Platform gibi hem API'leri 4 hem de bir UI'ı 2 olan sistemler için bu tür entegre testler özellikle değerlidir. Bu, UI'daki bir hatanın API'yi etkileyip etkilemediğini veya tam tersini anlamaya yardımcı olur, böylece hataların kaynağı daha hızlı tespit edilebilir.

#### Test Ortamı Tutarlılığı ve UI Testleri

Restful-Booker-Platform'un da (tıpkı Restful-Booker API'si gibi) her 10 dakikada bir sıfırlanması 2, UI testleri için önemli bir zorluk teşkil eder.

1. UI testleri, doğaları gereği API testlerinden genellikle daha yavaş çalışır. Bir UI testi, birden fazla sayfa geçişi, form doldurma ve eleman etkileşimi içerebilir.
2. Eğer bir UI test senaryosunun tamamlanması 10 dakikadan uzun sürerse, testin ortasında platformun sıfırlanmasıyla karşılaşılabilir.
3. Bu durum, testin başarısız olmasına, tutarsız sonuçlar üretmesine veya test sırasında kullanılan verilerin kaybolmasına neden olabilir. Bu sorunu yönetmek için UI test senaryoları daha kısa, modüler ve odaklanmış parçalara bölünmelidir. Alternatif olarak, test framework'ü içerisinde, platformun sıfırlanma durumunu algılayıp testi yeniden başlatan veya belirli bir kontrol noktasından devam ettiren mekanizmalar geliştirilebilir. Test başlangıcında ortamın durumunu kontrol etmek ve gerekirse test için gerekli önkoşulları yeniden oluşturmak da bir strateji olabilir.

## 6. Sonuç

### 6.1. Genel Test Değerlendirmesi

Restful-booker API'si üzerinde yürütülen testler, API'nin temel işlevlerinin büyük ölçüde çalıştığını göstermekle birlikte, bazı alanlarda iyileştirmelere ve dikkatli kullanıma ihtiyaç duyulduğunu ortaya koymuştur. API'nin birincil amacının API testi öğrenimi ve hataların keşfi için bir platform sağlamak olduğu göz önüne alındığında, karşılaşılan bazı "hatalar" aslında API'nin bu amacına hizmet eden kasıtlı özelliklerdir.

# 6.3. UI Testlerine Yönelik Değerlendirme

Automation in Testing platformunun kullanıcı arayüzü üzerinden yapılan UI testleri, rezervasyon oluşturma, tarih doğrulama ve paralel rezervasyon senaryolarının başarıyla çalıştığını ortaya koymuştur. Temel kullanıcı akışlarında (check-in ve check-out tarih seçimi, form doldurma ve rezervasyon işlemi) genel olarak sorunsuz bir kullanıcı deneyimi sağlandığı gözlemlenmiştir.

Ancak bazı önemli bulgular dikkat çekmiştir:

* **Hata mesajı görünürlüğü**: Check-out tarihi, check-in tarihinden önce seçildiğinde sistemin doğru şekilde hata vermesi beklenirken, uyarı mesajının bazı durumlarda kullanıcı tarafından fark edilmesi zor olabilmektedir. Bu durum, kullanıcı deneyimini olumsuz etkileyebilir.
* **Eş zamanlı rezervasyon girişimleri** (parallel booking): İki farklı thread ile aynı anda yapılan rezervasyon denemelerinde ikisinin de rezervasyon yapabildiği gözlemlenmiştir.   
  Bu durum, sistemde eşzamanlı işlem kontrolü (concurrency control) mekanizmasının olmadığını ya da yeterince uygulanmadığını göstermektedir.  
  Gerçek dünya senaryolarında, aynı odanın birden fazla kullanıcı tarafından aynı anda rezerve edilmesi veri tutarsızlığına yol açabileceğinden, back-end tarafında kilitleme (locking) ya da önceliklendirme gibi yöntemlerle bu durumun engellenmesi gerekir.
* **Form validasyonu**: E-posta gibi alanlara aynı değerle yapılan tekrar girişlerde beklenen şekilde hata mesajları verilmiştir. Ancak bu durum bazen uygulama hatası (ör. client-side exception) ile sonuçlanabilmektedir. Bu, daha sağlam form kontrolleri ve kullanıcı bilgilendirmesi ihtiyacını işaret etmektedir.

Sonuç olarak, UI testleri hem uygulamanın temel fonksiyonlarının doğruluğunu hem de kullanıcıya sunulan geri bildirimlerin yeterliliğini ortaya koymak açısından değerli bulgular sağlamıştır. API ile entegre şekilde çalışan arayüzde, doğrulama ve hata mesajlarının tutarlılığı ile kullanıcı yönlendirmesi açısından bazı küçük iyileştirmeler yapılması önerilmektedir.